Excel→PPT Agent (LangGraph + Gemini): Overview & Runbook

Give it an Excel file and a PowerPoint template; it plans/creates charts (optionally following your exact chart requirements), drops editable/native charts into named shapes in the PPT, and adds a short Insights section powered by Gemini.

# Graph structure (flow of steps)

Validate

↓

Load Mapping (optional YAML/JSON)

↓

Load Excel

↓

Plan Charts (Gemini, schema-only)

↓

Enforce Requirements (overlay your required chart specs)

↓

Render/Prepare Chart Payloads (data series + PNG fallback)

↓

Generate Insights (Gemini, aggregates-only)

↓

Bind to PPT (native editable charts + mapping to named shapes)

↓

End

# Each node — purpose & fallbacks

## 1) Validate

* Checks file paths exist (Excel, PPTX).
* Creates a temp folder for images (used as fallback).
* If a path is wrong → stops with a helpful error.

## 2) Load Mapping (optional)

* Reads --map YAML/JSON if provided.
* Expected keys (all optional):
* • charts: { <chart\_id>: <shape\_name> }
* • insights: <shape\_name>
* • requirements: either
* – simple: { <chart\_id>: 'line' | 'bar' }
* – explicit: { type, x, y, agg, top\_k }
* No mapping file? The agent still works.

## 3) Load Excel

* Reads the first sheet; best‑effort parses columns containing “date” as datetimes.
* Keeps data in memory (no uploads).

## 4) Plan Charts (Gemini)

* Sends only a schema/profile of the data (column names, basic stats, a few top values); no raw rows.
* Asks Gemini to suggest 1–2 charts as strict JSON (ChartPlan): id, type (line|bar), x, y, agg, top\_k.
* If the call fails, falls back to a sensible default (time‑series line or top‑K bar).

## 5) Enforce Requirements (your overlay)

* Applies optional requirements from mapping:
* • If you provide type only (e.g., ts: line), it picks suitable columns for you.
* • If you provide explicit columns (x/y/agg/top\_k), it validates and fills gaps with heuristics.
* Ensures required charts exist even if Gemini didn’t propose them.

## 6) Render/Prepare Chart Payloads

* Builds native chart payloads (categories, values, series\_name, type, title).
* Also saves a small PNG per chart as a last‑resort fallback.

## 7) Generate Insights (Gemini)

* Computes tiny aggregates locally (sum/mean for a few numeric columns) and sends those plus chart titles.
* Returns structured output: a 1–2 sentence summary + 3–5 bullets.
* If the model call fails, writes a friendly placeholder.

## 8) Bind to PPT (native, editable charts)

* For each mapped chart ID → look up the named shape anywhere in the template:
* 1) If the shape already contains a chart, replace its data (editable).
* 2) If it's a placeholder, insert a native chart into it.
* 3) Otherwise, add a native chart at the shape’s position/size.
* 4) If native fails, drop the PNG fallback.
* Unmapped charts get new slides with native charts.
* Insights write into the mapped text box; otherwise create a 'Key Insights' slide.
* Saves as <template\_basename>\_filled.pptx.

# Shared state (carried between steps)

* excel\_path, ppt\_template\_path — input paths
* mapping\_path — optional YAML/JSON file
* mapping — parsed charts↔shapes, insights target, and requirements
* df — DataFrame loaded from the Excel (first sheet)
* chart\_plan — final plan after Gemini + requirement overlay
* charts — prepared payloads for native PPT (and a PNG fallback path)
* insights — summary + bullets for the insights slide
* ppt\_out\_path — saved output file path
* log — human‑readable steps/fallbacks

# Inputs

## Required

* --excel path/to/data.xlsx
* --template path/to/template.pptx
* .env with GOOGLE\_API\_KEY=... (or GEMINI\_API\_KEY)

## Optional

* --map path/to/mapping.yaml (or .json)
* charts: { chart\_id: shape\_name }
* insights: TEXT\_insights
* requirements:
* • simple: { chart\_id: 'line' | 'bar' }
* • explicit: { type, x, y, agg, top\_k }

# Output

* A filled deck saved next to your template, e.g., template\_filled.pptx.
* Charts are editable/native unless the PNG fallback was needed.
* Console log shows requirement enforcement, missing shapes, and any fallbacks.

# Requirements (packages & openness)

* langchain-google-genai — Open‑source wrapper for Gemini in LangChain (service requires API key).
* google-genai — Open‑source SDK client; connects to proprietary Gemini API.
* langgraph — Open‑source orchestration for agent graphs.
* pandas — Open‑source data frames.
* numpy — Open‑source numerics.
* matplotlib — Open‑source plotting (used only to create PNG fallback images).
* python-pptx — Open‑source PowerPoint writer/editor (native charts).
* pydantic — Open‑source data validation / structured schemas.
* python-dotenv — Open‑source .env loader.
* pyyaml — Open‑source YAML parser (only if you use YAML mapping files).

## Install (one time)

pip install -U langchain-google-genai google-genai langgraph pandas numpy matplotlib python-pptx pydantic python-dotenv pyyaml

# How to run

## Minimal (no mapping)

python excel\_to\_ppt\_agent\_langchain\_env\_mapping\_requirements\_columns\_insights.py --excel ./data.xlsx --template ./template.pptx

## With mapping + requirements

python excel\_to\_ppt\_agent\_langchain\_env\_mapping\_requirements\_columns\_insights.py --excel ./data.xlsx --template ./template.pptx --map ./mapping.yaml

## Example mapping.yaml

charts:

revenue\_trend: CHART\_revenue\_trend

top\_customers: CHART\_top\_customers

insights: TEXT\_insights

requirements:

revenue\_trend:

type: line

x: date

y: revenue

agg: sum

top\_customers:

type: bar

x: customer

y: revenue

top\_k: 10

# Edge cases and fallbacks

* Required chart not supported by data (e.g., no datetime for a line): we synthesize a best‑effort version (line vs row index or bar) and log it.
* Mapping points to a shape that doesn’t exist: logged; we create a new slide and place a native chart.
* Mapped chart ID wasn’t produced: requirement overlay tries to produce it; if still unavailable, remaining charts are placed and the miss is logged.
* Native chart insertion fails: PNG fallback is used, so the deck still renders.

# Template tips

* Use the Selection Pane in PowerPoint to rename shapes where you want content.
* Name chart targets like CHART\_<id> (e.g., CHART\_revenue\_trend) and the insights box TEXT\_insights.
* Avoid grouped targets for placement; plain placeholders or shapes work best.